```python
import numpy as np
import matplotlib.pyplot as plt

def cart2pol(x, y):
    rho = np.sqrt(x**2 + y**2)
    phi = np.arctan2(y, x)
    return(rho, phi)

def pol2cart(rho, phi):
    x = rho * np.cos(phi)
    y = rho * np.sin(phi)
    return(x, y)

################################
#initial conditions

x_0 = 10
y_0 = -50
v_int = 1/10

r_0 = [(x_0, 0)]
v_0 = [(0, v_int)]

r = r_0
v = v_0

h = x_0*v_int
p = h**2
E_0 = .5*v_int**2-1/x_0
a = -1/(2*E_0)
e = (1-p/a)**.5
P = 2*np.pi*a**(3/2)
t = P/1000

##############################################
#Exact Solution

theta_ex = np.arange(0, 2*np.pi+.01, 0.01)
r_ex = p/(1-e*np.cos((theta_ex)))

theta_comp = np.arange(0, 2*np.pi+.01, 0.1)
r_comp = p/(1-e*np.cos((theta_comp)))



############################################################
#Graphing
fig=plt.figure(1)


#Exact Solution
ax1=fig.add_subplot(221, projection='polar')
ax1.plot(theta_ex, r_ex)
ax1.set_rmax(.5)
ax1.set_rticks([3, 6, 9, 12])  # less radial ticks
ax1.set_rlabel_position(-22.5)  # get radial labels away from plotted line
ax1.grid(True)
ax1.set_title("Kepler Solution", va='bottom')


##############################################
#Cromer Algorithm

for i in range(0,int(P*100)):
    r_mag = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
    acc = np.asarray((-r[len(r)-1][0]/r_mag**3,-r[len(r)-1][1]/r_mag**3))

```

```
        v.append(tuple(map(sum, zip(v[len(v)-1],acc*t))))
        v_add = np.asarray(v[len(v)-1])
        r.append(tuple(map(sum, zip(r[len(r)-1],v_add*t))))

    r = np.asarray(r)
    l = []

    for i in r:
        l.append(cart2pol(i[0],i[1]))

    x_val = [x[0] for x in l]
    y_val = [x[1] for x in l]



    ###############################################
    #Graphing Cromer Algorithm

    ax2=fig.add_subplot(222, projection='polar')
    ax2.plot(y_val,x_val)
    ax2.plot(theta_comp, r_comp, 'o', markerfacecolor='none', markeredgecolor='r')
    ax2.set_rmax(.5)
    ax2.set_rticks([3, 6, 9, 12])  # less radial ticks
    ax2.set_rlabel_position(-22.5)  # get radial labels away from plotted line
    ax2.grid(True)
    ax2.set_title("Cromer Algorithm", va='bottom')

    ##################################################
    #Cromer Energy

    r_mag = []
    v_mag = []
    E_t = []
    time = []
    timePeriod = []
    E_rat = []
    for i in r:
        r_mag.append((i[0]**2+i[1]**2)**.5)
    for i in v:
        v_mag.append((i[0]**2+i[1]**2)**.5)
    for i in range(len(r_mag)):
        E_t.append(.5*v_mag[i]**2-1/r_mag[i])
    for i in range(0,int(P*100)+1):
        x = i*t
        time.append(x)
    for i in time:
        x = i/P
        timePeriod.append(x)
    for i in range(len(E_t)):
        E_rat.append(E_t[i]/E_0-1)

    E_rat = E_rat[450:550]
    timePeriod = timePeriod[450:550]

    fig2, ax5 = plt.subplots()
    ax5.set_ylabel('E(t)/E_0-1')
    ax5.set_xlabel('Time/Period')
    ax5.set_title("Energy Ratio")
    ax5.plot(timePeriod,E_rat)

    ###############################################
    #Runge-Kutta
    x_0 = 10
    y_0 = -50
    v_int = 1/10

    r_0 = [(x_0, 0)]
```

```
v_0 = [(0, v_int)]

r = r_0
v = v_0

for i in range(0,int(P*100)):
    r_mag = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
    acc = np.asarray((-r[len(r)-1][0]/r_mag**3,-r[len(r)-1][1]/r_mag**3))

    v_add = np.asarray(v[len(v)-1])
    r_add = np.asarray(r[len(r)-1])

    r.append(tuple(map(sum, zip(r[len(r)-1],v_add*t,.5*t**2*acc))))

    r_2 = r_add+.5*t*v_add
    r_mag2 = ((r_add[0]+.5*t*v_add[0])**2+(r_add[1]+.5*t*v_add[1])**2)**.5
    acc_2 = np.asarray((-r_2[0]/r_mag2**3,-r_2[1]/r_mag2**3))


    v.append(tuple(map(sum, zip(v[len(v)-1],t*acc_2))))

r = np.asarray(r)
l = []

for i in r:
    l.append(cart2pol(i[0],i[1]))

x_val = [x[0] for x in l]
y_val = [x[1] for x in l]

#Graphing

ax3=fig.add_subplot(223, projection='polar')
ax3.plot(y_val,x_val)
ax3.plot(theta_comp, r_comp, 'o', markerfacecolor='none', markeredgecolor='r')
ax3.set_rmax(.5)
ax3.set_rticks([3, 6, 9, 12])  # less radial ticks
ax3.set_rlabel_position(-22.5)  # get radial labels away from plotted line
ax3.grid(True)
ax3.set_title("Runge-Kutta", va='bottom')

####################################################
#Runge-Kutta Energy

r_mag = []
v_mag = []
E_t = []
time = []
timePeriod = []
E_rat = []
for i in r:
    r_mag.append((i[0]**2+i[1]**2)**.5)
for i in v:
    v_mag.append((i[0]**2+i[1]**2)**.5)
for i in range(len(r_mag)):
    E_t.append(.5*v_mag[i]**2-1/r_mag[i])
for i in range(0,int(P*100)+1):
    x = i*t
    time.append(x)
for i in time:
    x = i/P
    timePeriod.append(x)
for i in range(len(E_t)):
    E_rat.append(E_t[i]/E_0-1)

E_rat = E_rat[450:550]
timePeriod = timePeriod[450:550]
```

```
# fig2, ax5 = plt.subplots()
ax5.plot(timePeriod,E_rat)




##############################################
#Velocity Verlet
x_0 = 10
y_0 = -50
v_int = 1/10

r_0 = [(x_0, 0)]
v_0 = [(0, v_int)]

r = r_0
v = v_0


for i in range(0,int(P*100)):
    r_mag = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
    acc = np.asarray((-r[len(r)-1][0]/r_mag**3,-r[len(r)-1][1]/r_mag**3))
    v_add = np.asarray(v[len(v)-1])
    r_add = np.asarray(r[len(r)-1])

    r.append(tuple(map(sum, zip(r[len(r)-1],v_add*t,.5*t**2*acc))))

    r_mag2 = (r[len(r)-1][0]**2+r[len(r)-1][1]**2)**.5
    acc_2 = np.asarray((-r[len(r)-1][0]/r_mag2**3,-r[len(r)-1][1]/r_mag2**3))

    v.append(tuple(map(sum, zip(v[len(v)-1],.5*t*acc,.5*t*acc_2))))

r = np.asarray(r)
l = []

for i in r:
    l.append(cart2pol(i[0],i[1]))

x_val = []
y_val = []

x_val = [x[0] for x in l]
y_val = [x[1] for x in l]


#Graphing
ax4=fig.add_subplot(224, projection='polar')
ax4.plot(y_val,x_val)
ax4.plot(theta_comp, r_comp, 'o', markerfacecolor='none', markeredgecolor='r')
ax4.set_rmax(.5)
ax4.set_rticks([3, 6, 9, 12])  # less radial ticks
ax4.set_rlabel_position(-22.5)  # get radial labels away from plotted line
ax4.grid(True)
ax4.set_title("Velocity Verlet", va='bottom')



##################################################
#Velocity Verlet Energy

r_mag = []
v_mag = []
E_t = []
time = []
timePeriod = []
E_rat = []
for i in r:
```

```
        r_mag.append((i[0]**2+i[1]**2)**.5)
    for i in v:
        v_mag.append((i[0]**2+i[1]**2)**.5)
    for i in range(len(r_mag)):
        E_t.append(.5*v_mag[i]**2-1/r_mag[i])
    for i in range(0,int(P*100)+1):
        x = i*t
        time.append(x)
    for i in time:
        x = i/P
        timePeriod.append(x)
    for i in range(len(E_t)):
        E_rat.append(E_t[i]/E_0-1)

    E_rat = E_rat[450:550]
    timePeriod = timePeriod[450:550]
    # fig2, ax5 = plt.subplots()
    ax5.plot(timePeriod,E_rat)

    plt.legend(('Cromer', 'Runge-Kutta', 'Velocity Verlet'), loc='upper right')
    plt.show()
    ################################################################
```